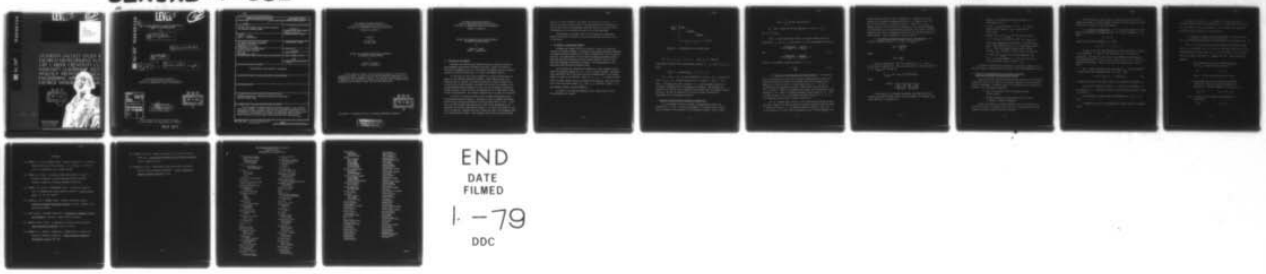
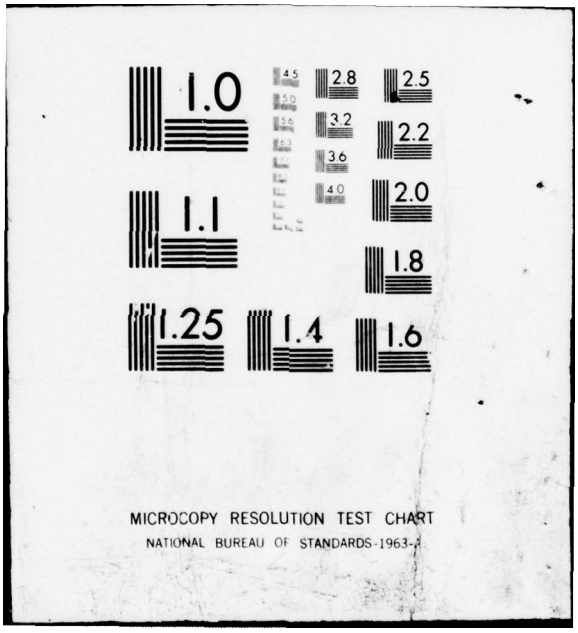
AD-A060 788 GEORGE WASHINGTON UNIV WASHINGTON D C PROGRAM IN LOG--ETC F/G 9/2
OPTIMAL TIME INTERVALS FOR TESTING HYPOTHESIS ON COMPUTER SOFTW--ETC(U)
JUN 78 E H FORMAN, N D SINGPURWALLA N00014-75-C-0729

UNCLASSIFIED SERIAL-T-382 NL

1 OF 1
AD
A060788



END
DATE
FILMED
1-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL II

THE GEORGE WASHINGTON UNIVERSITY

STUDENTS FACULTY STUDY R ESEARCH DEVELOPMENT FUT URE CAREER CREATIVITY CO MMUNITY LEADERSHIP TECH NOLOGY FRONTIE ESIGN ENGINEERING APP ENC GEORGE WASHIN NIV

INSTITUTE FOR MANAGEMENT SCIENCE AND ENGINEERING
SCHOOL OF ENGINEERING AND APPLIED SCIENCE

LEVEL II

(12)

(6) OPTIMAL TIME INTERVALS FOR TESTING HYPOTHESIS ON COMPUTER SOFTWARE ERRORS

by

(10) Ernest H. Forman
Nozer D. Singpurwalla

(14) Serial-T-382

(11) Serial T-382
26 June 1978

(12) 18 p. (9) Scientific rept.

The George Washington University
School of Engineering and Applied Science
Institute for Management Science and Engineering

(13) Program in Logistics
Contract N00014-75-C-0729
Project NR 347 020
Office of Naval Research

405 337

NONE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER T-382	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) OPTIMAL TIME INTERVALS FOR TESTING HYPOTHESIS ON COMPUTER SOFTWARE ERRORS		5. TYPE OF REPORT & PERIOD COVERED SCIENTIFIC 6. PERFORMING ORG. REPORT NUMBER N00014-75-C-0729
7. AUTHOR(s) ERNEST H. FORMAN NOZER D. SINGPURWALLA		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS THE GEORGE WASHINGTON UNIVERSITY PROGRAM IN LOGISTICS WASHINGTON, D.C. 20037		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS OFFICE OF NAVAL RESEARCH CODE 430D ARLINGTON, VA 22217		12. REPORT DATE 26 JUNE 13. NUMBER OF PAGES 14
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) NONE 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

DISTRIBUTION OF THIS REPORT IS UNLIMITED.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

SOFTWARE RELIABILITY, DEBUGGING PROCEDURE, STOPPING RULE, OPTIMAL TESTING TIME.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this paper we discuss certain stochastic aspects of the software reliability problem. We shall first discuss an empirical stopping note for debugging and testing computer software. We shall next present some results on choosing a time interval for testing the hypothesis that a software system contains no errors, given certain cost and risk constraints.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

NONE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

THE GEORGE WASHINGTON UNIVERSITY
School of Engineering and Applied Science
Institute for Management Science and Engineering
Program in Logistics

Abstract
of
Serial T-382
26 June 1978

OPTIMAL TIME INTERVALS FOR TESTING HYPOTHESIS ON COMPUTER SOFTWARE ERRORS

by

Ernest H. Forman*
Nozer D. Singpurwalla

In this paper we discuss certain stochastic aspects of the software reliability problem. We shall first discuss an empirical stopping note for debugging and testing computer software. We shall next present some results on choosing a time interval for testing the hypothesis that a software system contains no errors, given certain cost and risk constraints.

*Department of Management Science, The George Washington University

THE GEORGE WASHINGTON UNIVERSITY
School of Engineering and Applied Science
Institute for Management Science and Engineering
Program in Logistics

OPTIMAL TIME INTERVALS FOR TESTING HYPOTHESIS ON COMPUTER SOFTWARE ERRORS

by

Ernest H. Forman
Nozer D. Singpurwalla

1. Introduction and Summary

Research into improving and measuring computer software reliability has progressed along several different directions. Typical of these are structured programming, proofs of correctness of programs, and the stochastic analysis of software failure data [cf. Amster and Shooman (1975)].

In this paper we shall focus attention on certain decision-theoretic aspects of the software reliability problem. These aspects arise quite naturally when we consider a statistical analysis of software failure data. In particular, we shall develop a procedure for testing the hypothesis that a given software system contains no errors, and in the sequel, determine an optimal interval of time for which the software has to be exercised in order to test this hypothesis. The overall organization of our paper is as follows.

In Section 2, we shall briefly review a simple probabilistic model for describing software failures. This model is due to Jelinski and Moranda (1972) and has also been described by Lloyd and Lipow (1977) p. 516. In Section 3 we shall present some results pertaining to the estimation of the parameters of the model discussed in Section 2. We shall also present in Section 3 an empirical stopping rule which signals the end of the debugging phase for a given software system. This stopping rule has been recently proposed

by us in the open literature [see Forman and Singpurwalla (1977)]. In Section 4 we shall discuss a test of the hypothesis that the software contains no more errors, and determine an optimal interval of time for which the program has to be exercised in order to test the hypothesis.

The material in Sections 2 and 3 can be regarded as expository, whereas the material in Section 4 is new and represents the raison d'etre of this paper.

2. The Model by Jelinski and Moranda

Jelinski and Moranda (1972) have proposed a model for describing failures of computer software. Variations of this model have been considered by Shooman and others [Shooman et al. 1972a; Shooman 1972b, 1973] in several contexts. The applicability of this model for analyzing software failure data from the Apollo program and from a certain system of the U.S. Navy have been discussed by Jelinski and Moranda. Other applications of this model have been described by Moranda (1975).

Let us denote the initial error content in a large software system, such as an operating system, by N ; N is, of course, unknown. By assumption, the failure rate at any point in time is proportional to the residual number of errors in the software. Thus, if τ_1 , τ_2 , ... , denote the time points at which software errors are detected and corrected, then the failure rate at any time point between τ_{i-1} and τ_i is $(N-i+1)\phi$, where ϕ is some unknown constant of proportionality.

In Figure 2.1 we show the behavior of the failure rate for this "de-eutrophication" process.

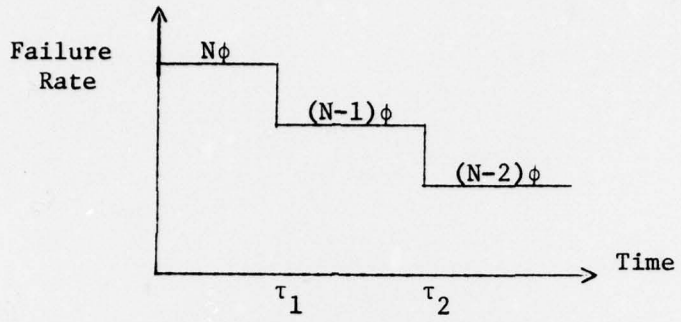


Figure 2.1. Time Behavior of the Failure Rate

Let $T_i = \tau_i - \tau_{i-1}$, $i = 1, 2, \ldots,$ with $\tau_0 \equiv 0$; then the distribution function of the times between failures T_i is $F(t_i) = P(T_i \leq t_i)$, where

$$F(t_i) = 1 - \exp(-(N-i+1)\phi) \ , \ \phi > 0 \ , \ t_i > 0 \ .$$

Let $t_1, t_2, \ldots, t_n$ be the realizations of $T_1, T_2, \ldots, T_n$, respectively. Given $t_1, t_2, \ldots, t_n$, a natural objective is to estimate ϕ and N . Note that n is always less than or equal to N . A more pragmatic objective is to obtain a "stopping rule" for debugging the software. The stopping rule should be such that we can be reasonably well assured that the software contains no errors. As will be pointed out in the next section, the above two objectives will be concurrently achieved.

3. Parameter Estimation and an Empirical Stopping Rule

As is discussed in detail by Forman and Singpurwalla (1977), the estimation of N and ϕ , and the development of a stopping rule are based upon an analysis of the behavior of the likelihood function $L(\phi, N)$, where

$$L(\phi,N) = \prod_{i=1}^{n} (N-i+1)\phi \exp(-(N-i+1)\phi) .$$

Let $L(\phi,N)$ denote the natural logarithm of $L(\phi,N)$, $T = \sum_{i}^{n} t_i$ and $k = \sum_{i}^{n} (i-1)t_i$.

Since N takes only integer values, the unique maximum likelihood estimator of N, say $\hat{N}$, is that value of N which simultaneously satisfies

$$n \log\left[\frac{(N-1)T-k}{TN-k}\right] + \log\left(\frac{N}{N-n}\right) \geq 0$$

and (3.1)

$$n \log\left[\frac{(N+1)T-k}{TN-k}\right] + \log\left(\frac{N-n+1}{N+1}\right) \geq 0 .$$

Given $\hat{N}$, the maximum likelihood estimator of ϕ, is

$$\hat{\phi} = \frac{n}{\hat{T}N-k} . \qquad (3.2)$$

3.1 Properties of the Maximum Likelihood Estimators. The estimators given by Equations (3.1) and (3.2) are quite straightforward to obtain, and have been discussed by Jelinski and Moranda (1972) and by Shooman (1973). However, the fact that when n is much smaller than N, $\hat{N}$ is *highly misleading* has been grossly overlooked. Specifically, when the quantity k/T is small, $\hat{N}$ tends to be unrealistically large, and furthermore, a slight decrease in k/T leads to a disproportionately large increase in $\hat{N}$. Thus, for small values of k/T, the maximum likelihood estimator of N, $\hat{N}$ is *very unstable*, and may lead to *erroneous conclusions*.

As k/T becomes large, that is, if the times between failures during the latter stages of testing are greater than those during the earlier stages, $\hat{N}$ tends to be close to n, the observed number of failures. Thus $\hat{N} \approx n$ is an indicator of the fact that the program is close to being debugged, and

should therefore provide us with a stopping rule. However, as has been pointed out by Forman and Singpurwalla (1977), it is possible that $\hat{N} \approx n$ and yet the true value N may be far from $\hat{N}$. Thus $\hat{N} \approx n$ is not always conclusive of the fact that the program is close to being debugged and, therefore, has to be interpreted with caution. For a more conclusive analysis we will have to examine the behavior of the likelihood function in greater detail. This is discussed in the next section.

3.2 The Relative Likelihood Functions and a Stopping Rule. The *relative likelihood function* of N, $R(N)$ is defined as

$$R(N) = \frac{L(N,\hat{\phi}(N))}{L(\hat{N},\hat{\phi})}$$

where

$$\hat{\phi}(N) = \frac{n}{TN-k} .$$

Note that the shape of $R(N)$ is a function of n, k, and T. We will need to compare $R(N)$ with the *normal relative likelihood function* of N, $R_{normal}(N)$ defined as

$$R_{normal}(N) = \exp\left[-\frac{1}{2}(\hat{N}-N)^2/Var(\hat{N})\right]$$

where

$$Var(\hat{N}) = \frac{n}{n\sum_{i}^{n}\left(\frac{1}{\hat{N}-i+1}\right)^2 - \left(\sum_{i}^{n}\frac{1}{\hat{N}-i+1}\right)^2} .$$

In the light of our previous discussions, plus some Monte Carlo analyses performed by Forman (1974), we shall propose the following steps which constitute a stopping rule.

1. Compute $\hat{N}$, the maximum likelihood estimator of N using Equation (3.1).

2. If $\hat{N} \approx n$, proceed to Step 3; if $\hat{N} >> n$, observe another failure interval t_{n+1} , and go back to Step 1 above.

3. Compute $R(N)$ and $R_{normal}(N)$ for various values of N , and see if the plots of $R(N)$ and $R_{normal}(N)$ are in good agreement with each other. If the two plots show a large disparity, then $\hat{N}$ is a misleading estimator of N . When this happens, observe another failure interval t_{n+1} and repeat the above steps. If the plots of $R(N)$ and $R_{normal}(N)$ show good agreement, then $\hat{N} \approx n$ is a good estimator of N and we do not have to test the software further to obtain t_{n+1} .

An example illustrating the application of the above stopping rule to some real data on software failures is given in Forman and Singpurwalla (1977).

4. A Test of the Hypothesis that the Software Contains No Errors and an Optimal Time Interval for Testing

Let us assume that a given software has been subjected to the debugging process and that all the steps of our proposed stopping rule have been satisfactorily undertaken. A potential user of this program will be interested in answers to the following two questions.

1. What is the assurance that the program contains no more errors?

2. How much additional testing should be done in order to achieve a specified assurance?

Clearly, an answer to the above questions will be a function of the risks that the user is willing to take, the cost of testing, and the consequences of software failure during its operation.

In what follows, we shall attempt to answer the above questions in a quantitative manner. We shall attempt to answer the first question by formulating it as a problem of testing hypothesis.

4.1 Testing of the Hypothesis. Let N^* denote the number of errors which are remaining in a debugged program. Let our null hypothesis be H_0, where

$$H_0 : N^* = 0 ,$$

versus the alternative hypothesis H_1, where

$$H_1 : N^* = r , \quad r = 1, 2, \ldots, .$$

In order to test the above hypotheses, we shall exercise the software for an additional t_a units of time, and reject H_0 if a failure is encountered. Note that for such a test, the probability of rejecting the null hypothesis when it is true, is zero, since , when $N^* = 0$, we will not encounter any failures. Thus, for our test the so-called *Type I error is zero.*

Let β denote the power of our test; that is, β is the probability of rejecting the null hypothesis when it is false. Since

$$\beta = P[T \leq t_a | N = r] = 1 - \exp(-\phi r t_a) , \qquad (4.1)$$

the power of the test can be made as large as is desired by increasing t_a.

If the user is willing to specify a β, then we can calculate t_a by choosing $r = 1$. When this is done, our test procedure will have a power of *at least* β.

For convenience, we shall denote the dependence of t_a on β by $t_a(\beta)$.

To summarize, our test of hypothesis will proceed along the following lines.

The user will specify a β, the power of the test, and given β, we shall determine $t_a(\beta)$ using Equation (A.1). We shall test the software for $t_a(\beta)$ period of time, and accept H_0 if no failures are encountered during that period; otherwise, we shall reject it.

4.2 Choosing t_a Based on Cost Considerations. We can also choose the t_a discussed above based on cost considerations and the mission time t_m.

Let C_1 be the cost per unit time for testing the software; suppose that C_1 is a constant. Let C_2 be the cost incurred by the failure of the software during the mission time t_m; C_2 is also assumed to be a constant. Later on, we shall assume that C_1 changes with time. Three outcomes are possible:

i. The software fails during the additional testing time t_a, in which case the total cost is

$$C_1 t \qquad 0 \leq t \leq t_a$$

where t is the time at which failure occurs;

ii. The software does not fail during the additional testing time, but fails during its operation at some time t; when this happens, the total cost is

$$C_1 t_a + C_2 \qquad t_a \leq t \leq t_a + t_m ;$$

iii. No failure of the software is encountered; the total cost is

$$C_1 t_a \qquad t_a + t_m < t .$$

The total expected cost is therefore

$$E(C) = \int_0^{t_a} C_1 t \phi e^{-\phi t} dt + \int_{t_a}^{t_a + t_m} (C_1 t_a + C_2) \phi e^{-\phi t} dt$$

$$+ \int_{t_a + t_m}^{\infty} C_1 t_a \phi e^{-\phi t} dt \ .$$

In order to solve for t_a, we shall minimize $E(C)$. Since

$$\frac{d}{dt_a} E(C) = e^{-\phi t_a} \left[C_1 - \phi C_2 \left(1 - e^{-\phi t_m} \right) \right]$$

we claim that when $C_1 > \phi C_2 \left(1 - e^{-\phi t_m} \right)$, the value of t_a at which $E(C)$ is minimized is zero. That is, when the cost of testing is much larger than the cost of an operational failure, no additional testing is necessary. However, a potential user may still wish to test the program for $t_a(\beta)$ units of time and be assured that the power of his test is at least β.

If, on the contrary, $C_1 < \phi C_2 \left(1 - e^{-\phi t_m} \right)$, then $t_a = \infty$ minimizes $E(C)$. This means that we should test exercise the software for an indefinite amount of time. Here, again, a potential user may test for $t_a(\beta)$ units of time and get an assurance that the power of his test is at least β.

The assumption that C_1 is a constant may not be realistic in many situations. In addition to this, this assumption may lead us to the case of indefinite testing as is shown above. We shall now relax this assumption and explore the consequences.

Suppose that the cost of testing is $C_1(t)$, where $C_1(t)$ is a convex non-decreasing function of t. Let us denote the derivative of $C_1(t)$ evaluated at $t = 0$ by $C_1'(0)$.

We can now verify that when

$$C_1'(0) \geq \phi C_2\left(1-e^{-\phi t_m}\right)$$

$t_a = 0$ will minimize $E(C)$. If this happens, we can choose $t_a(\beta)$ as our additional test time.

If on the contrary

$$C_1'(0) < \phi C_2\left(1-e^{-\phi t_m}\right)$$

then $t_a = C_1'^{-1}\left[\phi C_2\left(1-e^{-\phi t_m}\right)\right]$ will minimize $E(C)$; $C_1'^{-1}(\cdot)$ is the inverse of $C_1'(\cdot)$.

As an example, if $C_1(t) = C_1 e^{\alpha t}$, for some $\alpha > 0$, then,

$$t_a = \frac{1}{\alpha} \log\left[\frac{C_2\phi\left(1-e^{-\phi t_m}\right)}{C_1\alpha}\right] .$$

For the above situation, if $t_a \geq t_a(\beta)$, and should we test for t_a units of time, then we will not only be minimizing the total expected cost, but will also achieve a power of at least β . If $t_a < t_a(\beta)$, then $C_1\left(e^{\alpha t_a(\beta)} - e^{\alpha t_a}\right)$ represents the increase in the cost of testing to achieve a power of at least β .

REFERENCES

[1] AMSTER, S. J. and M. SHOOMAN (1975). Software reliability: an overview. Reliability and fault tree analysis. (R. E. Barlow, J. B. Fussell and N. D. Singpurwalla, eds.) SIAM 655-685.

[2] FORMAN, E. H. (1974). Statistical models and methods for measuring software reliability. Technical Memorandum Serial TM-64805, Program in Logistics, The George Washington University.

[3] FORMAN, E. H. and N. D. SINGPURWALLA (1977). An empirical stopping rule for debugging and testing computer software. J. Amer. Statist. Assoc. 72 No. 360 750-757.

[4] JELINSKI, Z. and P. MORANDA (1972). Software reliability research. Statistical Computer Performance Evaluation 465-484 Academic Press, New York and London.

[5] LLOYD, DAVID K. and MYRON LIPOW (1977). Reliability: Management Methods and Mathematics (2nd ed.) Prentice Hall, New Jersey.

[6] MORANDA, PAUL B. (1975). A comparison of software error-rate models. Texas Conference on Computing 2A-6.1 - 2A.6.9.

[7] SHOOMAN, M., J. HESSE, A. KIENTZ and J. DICKSON (1972). Quantitative analysis of software reliability. Annual Reliability Symposium Proceedings, I.E.E.E., New York.

[8] SHOOMAN, M. (1972). Probabilistic models for software reliability prediction. International Symposium on Fault-Tolerant Computing, I.E.E.E. Computer Society.

[9] SHOOMAN, M. (1973). Operational testing and software reliability estimation during program development. I.E.E.E. Symposium on Computer Software Reliability, 51-57.

THE GEORGE WASHINGTON UNIVERSITY
Program in Logistics
Distribution List for Technical Papers

The George Washington University
- Office of Sponsored Research
- Library
- Vice President H. F. Bright
- Dean Harold Liebowitz
- Mr. I. Frank Doubleday

ONR
- Chief of Naval Research
  - (Codes 200, 430D, 1021P)
  - Resident Representative

OPNAV
- OP-40
- DCNO, Logistics
- Navy Dept Library
- OP-911
- OP-964

Naval Aviation Integrated Log Support

NAVCOSSACT

Naval Cmd Sys Sup Activity Tech Library

Naval Electronics Lab Library

Naval Facilities Eng Cmd Tech Library

Naval Ordnance Station
- Louisville, Ky.
- Indian Head, Md.

Naval Ordnance Sys Cmd Library

Naval Research Branch Office
- Boston
- Chicago
- New York
- Pasadena
- San Francisco

Naval Research Lab
- Tech Info Div
- Library, Code 2029 (ONRL)

Naval Ship Engng Center
- Philadelphia, Pa.
- Hyattsville, Md.

Naval Ship Res & Dev Center

Naval Sea Systems Command
- Tech Library
- Code 073

Naval Supply Systems Command
- Library
- Capt W. T. Nash

Naval War College Library
- Newport

BUPERS Tech Library

FMSO

Integrated Sea Lift Study

USN Ammo Depot Earle

USN Postgrad School Monterey
- Library
- Dr. Jack R. Borsting
- Prof C. R. Jones

US Marine Corps
- Commandant
- Deputy Chief of Staff, R&D

Marine Corps School Quantico
- Landing Force Dev Ctr
- Logistics Officer

Armed Forces Industrial College

Armed Forces Staff College

Army War College Library
- Carlisle Barracks

Army Cmd & Gen Staff College

US Army HQ
- LTC George L. Slyman
- Army Trans Mat Command

Army Logistics Mgmt Center
- Fort Lee

Commanding Officer, USALDSRA
- New Cumberland Army Depot

US Army Inventory Res Ofc
- Philadelphia

HQ, US Air Force
- AFADS 3

Griffiss Air Force Base
- Reliability Analysis Center

Maxwell Air Force Base Library

Wright-Patterson Air Force Base
- HQ, AF Log Command
- Research Sch Log

Defense Documentation Center

National Academy of Science
- Maritime Transportation Res Board Library

National Bureau of Standards
- Dr E. W. Cannon
- Dr Joan Rosenblatt

National Science Foundation

National Security Agency

WSEG

British Navy Staff

Logistics, OR Analysis Establishment
- National Defense Hdqtrs, Ottawa

American Power Jet Co
- George Chernowitz

ARCON Corp

General Dynamics, Pomona

General Research Corp
- Dr Hugh Cole
- Library

Planning Research Corp
- Los Angeles

Rand Corporation
- Library

Carnegie-Mellon University
- Dean H. A. Simon
- Prof G. Thompson

Case Western Reserve University
- Prof B. V. Dean
- Prof John R. Isbell
- Prof M. Mesarovic
- Prof S. Zacks

Cornell University
- Prof R. E. Bechhofer
- Prof R. W. Conway
- Prof J. Kiefer
- Prof Andrew Schultz, Jr.

Cowles Foundation for Research
- Library
- Prof Herbert Scarf
- Prof Martin Shubik

Florida State University
- Prof R. A. Bradley

Harvard University
- Prof K. J. Arrow
- Prof W. G. Cochran
- Prof Arthur Schleifer, Jr.

New York University
- Prof O. Morgenstern

Princeton University
- Prof A. W. Tucker
- Prof J. W. Tukey
- Prof Geoffrey S. Watson

Purdue University
Prof S. S. Gupta
Prof H. Rubin
Prof Andrew Whinston

Stanford
Prof T. W. Anderson
Prof G. B. Dantzig
Prof F. S. Hillier
Prof D. L. Iglehart
Prof Samuel Karlin
Prof G. J. Lieberman
Prof Herbert Solomon
Prof A. F. Veinott, Jr.

University of California, Berkeley
Prof R. E. Barlow
Prof D. Gale
Prof Rosedith Sitgreaves
Prof L. M. Tichvinsky

University of California, Los Angeles
Prof J. R. Jackson
Prof Jacob Marschak
Prof R. R. O'Neill
Numerical Analysis Res Librarian

University of North Carolina
Prof W. L. Smith
Prof M. R. Leadbetter

University of Pennsylvania
Prof Russell Ackoff
Prof Thomas L. Saaty

University of Texas
Prof A. Charnes

Yale University
Prof F. J. Anscombe
Prof I. R. Savage
Prof M. J. Sobel
Dept of Admin Sciences

Prof Z. W. Birnbaum
University of Washington

Prof B. H. Bissinger
The Pennsylvania State University

Prof Seth Bonder
University of Michigan

Prof G. E. P. Box
University of Wisconsin

Dr. Jerome Bracken
Institute for Defense Analyses

Prof H. Chernoff
MIT

Prof Arthur Cohen
Rutgers – The State University

Mr Wallace M. Cohen
US General Accounting Office

Prof C. Derman
Columbia University

Prof Paul S. Dwyer
Mackinaw City, Michigan

Prof Saul I. Gass
University of Maryland

Dr Donald P. Gaver
Carmel, California

Dr Murray A. Geisler
Logistics Mgmt Institute

Prof J. F. Hannan
Michigan State University

Prof H. O. Hartley
Texas A & M Foundation

Mr Gerald F. Hein
NASA, Lewis Research Center

Prof W. M. Hirsch
Courant Institute

Dr Alan J. Hoffman
IBM, Yorktown Heights

Dr Rudolf Husser
University of Bern, Switzerland

Prof J. H. K. Kao
Polytech Institute of New York

Prof W. Kruskal
University of Chicago

Prof C. E. Lemke
Rensselaer Polytech Institute

Prof Loynes
University of Sheffield, England

Prof Steven Nahmias
University of Pittsburgh

Prof D. B. Owen
Southern Methodist University

Prof E. Parzen
State University New York, Buffalo

Prof H. O. Posten
University of Connecticut

Prof R. Remage, Jr.
University of Delaware

Dr Fred Rigby
Texas Tech College

Mr David Rosenblatt
Washington, D. C.

Prof M. Rosenblatt
University of California, San Diego

Prof Alan J. Rowe
University of Southern California

Prof A. H. Rubenstein
Northwestern University

Dr M. E. Salveson
West Los Angeles

Prof Edward A. Silver
University of Waterloo, Canada

Prof R. M. Thrall
Rice University

Dr S. Vajda
University of Sussex, England

Prof T. M. Whitin
Wesleyan University

Prof Jacob Wolfowitz
University of Illinois

Mr Marshall K. Wood
National Planning Association

Prof Max A. Woodbury
Duke University